

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

First Named

Inventor : Timothy J. Bloch

Appln. No. : 10/081,921

Filed : February 20, 2002

Title : SYSTEM AND METHOD FOR  
DEPLOYING AND IMPLEMENTING  
SOFTWARE APPLICATION OVER A  
DISTRIBUTED NETWORK

Group Art Unit: 2193

Examiner:  
Mark P. Francis

Docket No. : J267.12-0001

**EXPRESS MAIL COVER SHEET**

Commissioner For Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

**SENT VIA EXPRESS MAIL**

Express Mail No.: EV 884205626 US

The following papers are being transmitted via **EXPRESS MAIL** to the U.S. Patent and Trademark Office on the date shown below:

1. Fee Transmittal (in duplicate) with attached check for \$250.00;
2. Transmittal of Appeal Brief; and
3. Appeal Brief (in triplicate).

Respectfully submitted,

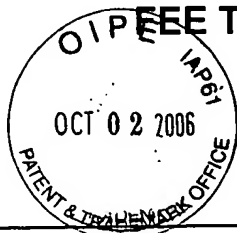
KINNEY & LANGE, P.A.

Date: 10/2/06

By

Michael A. Collins, Reg. No. 59,135  
THE KINNEY & LANGE BUILDING  
312 South Third Street  
Minneapolis, MN 55415-1002  
Telephone: (612) 339-1863  
Fax: (612) 339-6580

MAC:owk



# FEE TRANSMITTAL

## Complete if Known

Application No. 10/081,921  
Filing Date February 20, 2002  
First Named Inventor Timothy J. Bloch  
Group Art Unit 2193  
Examiner Name Mark P. Francis  
Atty. Docket Number J267.12-0001

Total Amount of Payment \$ 250.00

### METHOD OF PAYMENT (Check One)

1. ☒ The Commissioner is hereby authorized to charge any additional fee required under 37 C.F.R. 1.16 and 1.17 and credit any over payments to Deposit Account No. 11-0982. Deposit Account Name: Kinney & Lange, P.A. A duplicate copy of this communication is enclosed.

2. ☒ Check Enclosed

### FEE CALCULATION

#### 1. BASIC FILING FEE

Appn. Type	FILING FEE FEE/SMALL	SEARCH FEES FEE/SMALL	EXAM FEES FEE/SMALL	FEES
Utility	300 / 150	500 / 250	200 / 100	—
Design	200 / 100	100 / 50	130 / 65	—
Reissue	300 / 150	500 / 250	600 / 300	—
Provisional	200 / 100	-0- / -0-	-0- / -0-	—

Subtotal (1) \$-0-

#### 2. EXTRA CLAIM FEES

	Number Claims	Prior	Extra	Fee from Below	Fee Paid
Total	21	=	0	x	25 = 0
Indep.	3	=	0	x	100 = 0

Multiple Dependent Claims — = —

Insert 3 and 20, or number previously paid if greater; Reissue see below

Large Entity Fee Code	Fee (\$)	Small Entity Fee Code	Fee (\$)	Description
1202	50	2202	25	Claims in excess of 20
1201	200	2201	100	Independent claims in excess of 3
1203	360	2203	180	Multiple Dependent Claim
1204	200	2204	100	Reissue Independent Claims Over Original Patent
1205	50	2205	25	Reissue claims in excess of 20 and over original patent

#### 3. APPLICATION SIZE FEE

If the specification and drawings exceed 100 sheets of paper, the application size fee due is \$250 (\$125 small) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 C.F.R. 1.16(s). \$-0-

Subtotal (2) \$-0-

### FEE CALCULATION (Continued)

#### 3. ADDITIONAL FEES

Large Entity Fee Code	Fee (\$)	Small Entity Fee Code	Fee (\$)	Fee Description	Fee paid
1051	130	2051	65	Surcharge - Late filing fee or oath	—
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet	—
1053	130	1053	130	Non-English specification	—
1812	2,520	1812	2,520	For Filing a Request for Reexamination	—
1251	120	2251	60	Extension for reply within first month	—
1252	450	2252	225	Extension for reply within second month	—
1253	1,020	2253	510	Extension for reply within third month	—
1254	1,590	2254	795	Extension for reply within fourth month	—
1255	2,160	2255	1,080	Extension for reply within fifth month	—
1402	500	2402	250	Filing a brief in support of an appeal	250
1403	1,000	2403	500	Request for oral hearing	—
1814	130	2814	65	Terminal Disclaimer Fee	—
1452	500	2452	250	Petition to revive - unavoidable	—
1453	1,500	2453	750	Petition to revive - unintentional	—
1501	1,400	2501	700	Utility/Reissue issue fee	—
1502	800	2502	400	Design issue fee	—
1460	130	1460	130	Petitions to the Commissioner	—
1807	50	1807	50	Petitions related to provisional applications	—
1806	180	1806	180	Submission of Information Disclosure Statement	—
8021	40	8021	40	Recording each patent assignment per property (times number of properties)	—
1801	790	2801	395	Request for Continued Examination (RCE)	—
Other fee (specify) _____					—

Subtotal (3) \$250.00

Signature Michael A. Collins

Reg. No. 59,135

Date 10/2/06

Deposit Account No. 11-0982

10-03-06  
IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

First Named

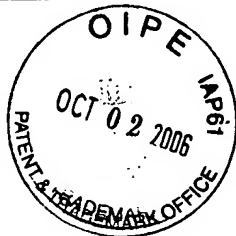
Inventor : Timothy J. Bloch

Appln. No. : 10/081,921

Filed : February 20, 2002

Title : SYSTEM AND METHOD FOR DEPLOYING AND  
IMPLEMENTING SOFTWARE APPLICATION OVER A  
DISTRIBUTED NETWORK

Docket No. : J267.12-0001



Appeal No.

Group Art Unit: 2193

Examiner: Mark P. Francis

**TRANSMITTAL OF APPEAL BRIEF  
(PATENT APPLICATION - 37 C.F.R. 1.192)**

Commissioner For Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

**SENT VIA EXPRESS MAIL**  
Express Mail No.: EV 884205626 US

Transmitted herewith in triplicate is the Appeal Brief in this application with respect to the  
Notice of Appeal filed on July 13, 2006.

**Fee Status**

☒ Small entity status under 37 C.F.R. 1.9 and 1.27 is established.

**Filing Fee**

☒ Pursuant to 37 C.F.R. 1.17(c) the fee of \$250.00 for filing the Appeal Brief was  
submitted on October 2, 2006.

Respectfully submitted,

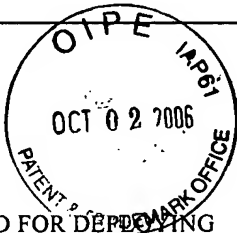
KINNEY & LANGE, P.A.

By: Michael A. Collins

Michael A. Collins, Reg. No. 59,135  
THE KINNEY & LANGE BUILDING  
312 South 3rd Street  
Minneapolis, MN 55415-1002  
Telephone: (612) 339-1863  
Fax: (612) 339-6580

MAC:owk

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

First Named Inventor	:	Timothy J. Bloch			
Appln. No.	:	10/081,921			
Filed	:	February 20, 2002			Group Art Unit: 2193
Title:		SYSTEM AND METHOD FOR DEPENDENT AND IMPLEMENTING SOFTWARE APPLICATION OVER A DISTRIBUTED NETWORK			Examiner: Mark P. Francis
Docket No.	:	J267.12-0001			

**BRIEF FOR APPELLANT**

Commissioner For Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

**SENT VIA EXPRESS MAIL**  
Express Mail No.: EV884205626US

Sir:

Applicant appeals to the Board of Patent Appeals and Interferences from the decision of the Primary Examiner mailed June 23, 2006, finally rejecting claims 1-17 and 19-22.

**Real Party in Interest**

The real party in interest is Jargon Software of Minneapolis, MN who is the owner of the entire right, title, and interest in the application.

**Related Appeals and Interferences**

There are no known related appeals or interferences which will directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

**Status of the Claims**

- |     |   |   |
|-----|---|---|
| I.  | Total number of claims in the application |   |
|     | A.  | Claims in the application are: 1-22, inclusive. |
| II. | Status of all of the claims               |   |
|     | A.  | Claims canceled: 18                             |
|     | B.  | Claims withdrawn but not canceled: none         |

10/05/2006 RHEBRIGHT 00000031 10081921 250.00 0P  
01 FC:2402

C.	Claims pending:	1-17 and 19-22
D.	Claims allowed:	none
E.	Claims rejected:	1-17 and 19-22

### III. Claims on appeal

- A. The claims on appeal are: 1-17 and 19-22.

### Status of Amendments

An Amendment After Final was filed April 13, 2006, which was responsive to the Final Office Action, dated February 13, 2006. The Advisory Action, dated June 23, 2006, indicated that the arguments submitted by the Applicant were not persuasive. In the Amendment filed April 13, 2006, claim 18 was canceled and replaced by new claim 22. Although the Advisory Action did not indicate whether the Amendment was entered, it did refer to "claims 1-22" in the second line of page 2. Thus claims 1-17 and 19-22 are currently in the case.

### Summary of Claimed Subject Matter

The present invention, as set forth in independent claims 1, 8, and 16, is a system and method for deploying applications over a distributed network to an Internet-enabled device for interacting with a server. To deploy an application to the Internet-enabled device, the device downloads text files from the server. An application assembler, running on the Internet-enabled device, allows the Internet-enabled device to extract program logic from each of the downloaded text files and to assemble the retrieved program logic into a functioning application. The functioning application can then be run on the Internet-enabled device regardless of whether the device remains connected to the server.

This system is unique in the way applications are distributed to client devices. In particular, the application assembler (referred to in the specification as the Application Virtual Machine or "AVM") assembles a functioning application on a client device. Unlike prior art methods in which the application is run on the server as opposed to the client device, or in which executable code for an application must be downloaded to the client device, the

present invention *assembles a functioning application on the client device based on text files downloaded from a server*. Therefore, the present invention may distribute a number of different applications to client devices by providing a number of unique text files that may be downloaded and assembled by the Internet-enabled devices. Summaries of the independent claims of the present invention are provided below. The highlighted terms identify differences between the present invention and the prior art.

Independent claim 1 recites “**an application assembler for storing on and running on the Internet-enabled device, the application assembler for downloading one or more text files from the server, retrieving program logic from each of the downloaded text files, and assembling the retrieved program logic into a functioning application and running the functioning application on the Internet-enabled device** regardless of whether the Internet-enabled device remains connected to the server.”

Independent claim 8 requires “**a program assembler for storing on and running on the Internet-enabled device, the program assembler for downloading application logic files, retrieving embedded application logic from the application logic files, and building the computer program from the retrieved embedded application logic, and running the computer program on the Internet-enabled device**.”

Independent claim 16 requires “**storing and running a software module on a client device of a user, providing to the client device text files containing embedded application program logic for the software module, the text files containing embedded program logic for the computer program to the installed software module upon request; running the computer program assembled from the embedded program logic on the client device; and enabling user interaction with the computer program running on the client device**.”

### **Grounds of Rejection to be Reviewed on Appeal**

Claims 1-5 and 8-21 stand rejected under 35 U.S.C. § 102(e) as being anticipated by Rice III, U.S. Pub. 2002/0174010 (“the Rice publication”). Dependent claims 6 and 7 stand rejected under 35 U.S.C. § 103(a) as unpatentable over the Rice publication in view of Lloyd, U.S. Patent No. 6,779,178 (“the Lloyd patent”).

### Argument

In the Final Office Action of February 13, 2006, claims 1-5 and 8-21 were rejected under 35 U.S.C. § 102(e) as being anticipated by the Rice publication, and claims 6 and 7 were rejected under 35 U.S.C. § 103(a) as being unpatentable over the Rice publication in view of the Lloyd patent. In the Advisory Action of June 23, 2006, the rejection of claims 1-22 (as amended in the Amendment After Final of April 13, 2006) was reaffirmed, and Applicants' arguments were held to be unpersuasive.

#### I. Definition of Relevant Terms

The following definitions are provided for the more relevant terms appearing in the independent claims of the present application, the Rice publication, and the Lloyd patent.

The term "Internet-enabled device" is used in the present invention to describe a computer device that has the capability of connecting to a network. The Internet-enabled device is also described throughout the specification of present application as a "client device."

The terms "client" and "server" are used to identify the roles of respective computing devices in transactional relationships known as client/server interactions. The term "server" generally refers to a remote computer system that receives requests from a client, and that returns responses to the client over the network. The term client therefore refers to a local computer system that sends requests to a server, and that receives responses from the server over a network.

The term "text file" appears in the present invention and refers to a file that contains *human readable text*. This is in contrast with the term "executable code" that appears in both the present invention and the Rice publication, which is defined in the art as binary or *machine-readable code* that has been compiled and can be executed or run by the hardware of a particular device. For instance, the application assembler stored and run on the client device of the present invention is an example of executable code. However, the text files downloaded from the server to the client device in the present invention are not executable code because they cannot be understood or executed by the hardware of the client

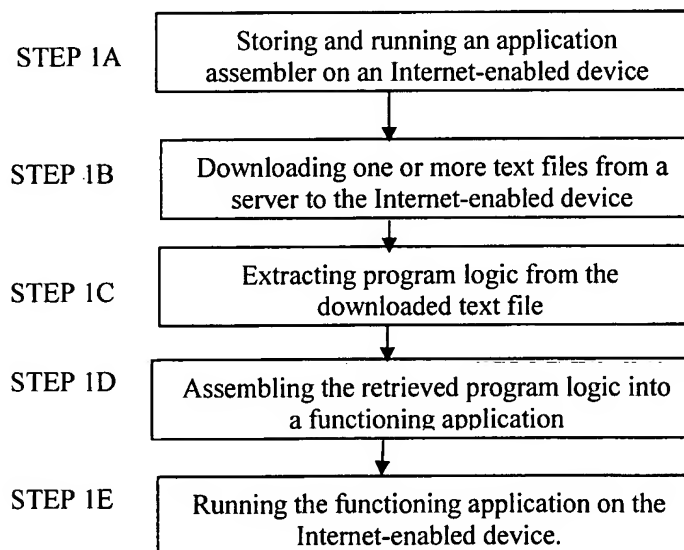
device.

The term “application assembler” refers to the application stored and run on the Internet-enabled device or client device of the present invention that assembles or builds a functioning application. The application assembler is comprised of executable code that runs on the client device and assembles or builds functioning applications based on input (e.g. text files containing embedded program logic) received from a server.

The term “assembling” refers to one of the functions performed by the application assembler, and consists of building a functioning application based on program logic retrieved from the text files provided by the server. Specifically, the application assembler uses the retrieved program logic, such as specifications of various visual components and scripts of actions to be taken in response to various user actions, and builds a functioning or “virtual” application. The functioning application is described in the specification as virtual because it does NOT require the compiling or download of executable code in order to run the assembled application. Therefore, the only executable code that is running on the client device of the present invention is the application assembler.

## II. Distribution of Applications in the Present Invention

The following flowchart illustrates the process of distributing an application to an Internet-enabled device (i.e., a client) using the system defined by the present invention..



**FIGURE 1 - OPERATION OF THE PRESENT INVENTION**



As shown in the above figure, the first step (step 1A) in distributing an application to an Internet-enabled device using the system described in independent claim 1 includes storing and running an application assembler on the Internet-enabled device. The application assembler is a program that is capable of assembling or generating a functional application based on program logic retrieved from a text file. The second step (step 1B) requires the Internet-enabled device to download one or more text files from a server. The term text file is used to describe a file that appears as plain text (i.e., human readable). An example of one type of text file that may be downloaded from the server is an XML document. (See Applicant's Specification page 28, line 13-16). In the third step (step 1C), the application assembler retrieves program logic from each of the downloaded text files. In the fourth step (step 1D), the application assembler uses the retrieved program logic to assemble a functioning application. That is, the application assembler uses the retrieved program logic, including specifications of various visual components and container objects that will be displayed to a user of the client device as well as scripts that specify actions to be taken at startup and upon recognizing various user actions, to present a functioning or "virtual" application to the user of the client device.

In the fifth step (step 1E), the functioning application is run on the Internet-enabled device, regardless of whether the client device is still connected to the server from which the text file was downloaded. Therefore, the system taught by the present invention does not rely on the server to run or execute the functioning application. Rather, the application assembler is capable of constructing the functioning application and running it locally on the Internet-enabled device (i.e., client). One of the benefits of this arrangement is the ability use and run the functioning application (following the download of the required text files from the server) without being connected to the server.

Furthermore, the present invention does not require the download or compilation of executable code in order to run the functioning application. Rather, the application assembler uses program logic retrieved from the text files to assemble the various visual components and container objects that will be displayed to a user of the client device and the scripts that specify the actions to be taken at startup and upon recognizing various user actions. The functioning application can then be run on the Internet-enabled device.

### III. Distribution of Applications in the Rice Publication

The system and method taught by independent claims 1, 8 and 16 for distributing applications to Internet-enabled devices (i.e. clients) is in contrast with the system and method described by the Rice publication. The Rice publication teaches two methods of distributing applications from a server to a client, neither of which is similar to the system or method used by the present invention. In the first method, the Rice publication describes a method of client/server interaction known as "thin-client". In the second method, the Rice publication describes a method of client/server interaction known as "fat/thick-client".

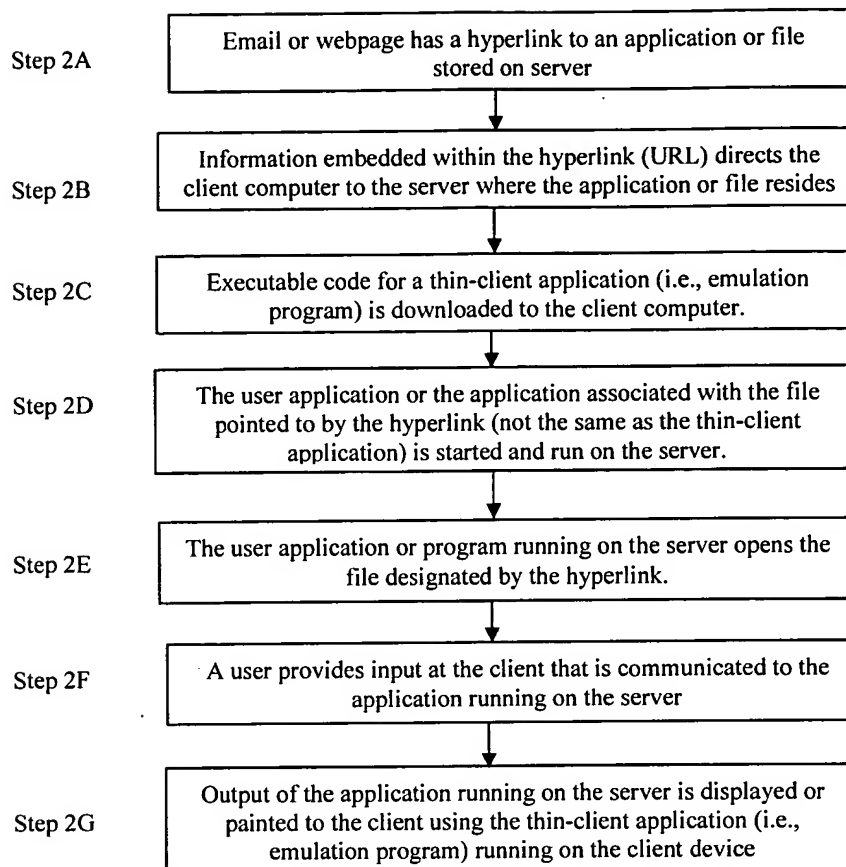
A thin-client mode<sup>1</sup>, as described in the Rice publication, is a client/server relationship in which all application execution takes place at the server and the client acts essentially as a terminal emulator that allows a user to submit input to the application running on the server and displays the output of the executable application running on the server. (See Paragraph 0008 of the Rice publication). In contrast, a fat/thick-client mode allows a client device to download executable code from a server such that the client device can execute the application independent of a connection between the client and the server. The Rice publication describes, in part, a device that allows a user to initially utilize a device in thin-client mode while the executable code for the application is being downloaded to the client device. Following the completion of the download of the executable code, the device (or client) is free to operate in fat/thick client mode in which communication with the server is no longer required. (See Paragraph 0014 of the Rice publication).

#### A. Thin-Client Mode Taught by the Rice Publication

More specifically, the thin-client mode of operation described by the Rice publication, and relied upon by the Examiner with respect to independent claims 1, 8 and 16 is described in paragraph 0107 of the Rice publication. The Applicant has provided the following flow chart to aid in the description of the operation of the thin-client mode taught by the Rice publication.

---

<sup>1</sup> The Applicant's specification also uses the term "thin-client" to describe the operation of the present invention. At the time the application was filed, no better term describing client/server interaction was available. Despite the similarity in terms, a number of differences exist between the operation of the



**FIGURE 2 - Thin-Client Mode of Operation Taught by Rice**

At step 2A, a user of the client device clicks on a hyperlink appearing in an email or webpage. As stated in Rice, “an email or a web page can have a hyperlink to a particular application file or document stored remotely on server 140 or remote machine 144.” (See paragraph 0107 of the Rice publication). At step 2B, “when the recipient or user using client computer 140 clicks on or executes the link, information embedded therein directs the client computer to a server computer 140 on which a data file and user application reside.” *Id.* That is, the *information embedded within the hyperlink acts as an address (e.g. URL)* that tells the client computer the location of the server where the data file or application is located.

At step 2C, “a thin-client [application] is automatically downloaded to the

---

present invention and the thin-client mode described by the Rice publication.

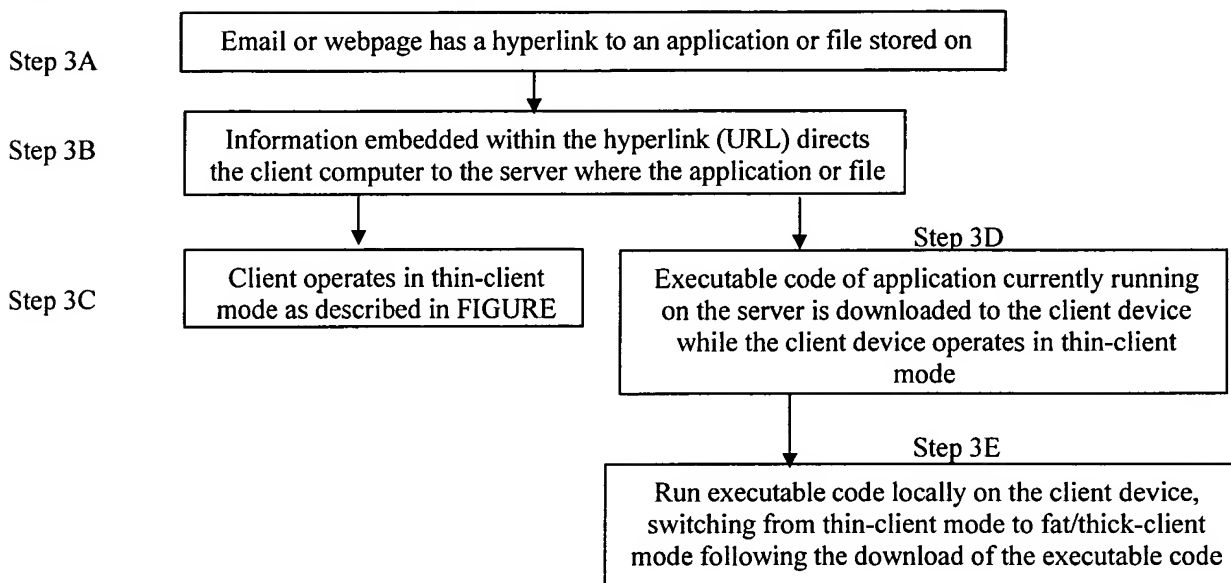
user.” Id. As described in the Rice publication, “upon execution of a hyperlink request for a document by a remote user, the user may be presented with a thin client *executable code* which permits viewing and editing of the data file in its native format ....” (Paragraph 0097 of the Rice publication). Therefore, the thin-client application is downloaded to the client device as *executable code* that is capable of being run by the client device. Preferably, the thin client application downloaded from the server to the client is an emulation program. (See Paragraph 0016 of the Rice publication).

At step 2D, “the application or program associated with the file [pointed to by the hyperlink] (typically the application that was used to created it, i.e., the data file’s native application) is started on a remote application server computer 142.” (Paragraph 0107 of the Rice publication). Therefore, the Rice publication makes clear that *the application runs on the server, not on the client device* during thin-client mode. At step 2E, “the specific file designated by the link is then opened by the user application” running on the server. Id.

At step 2F, the thin-client application or emulation program allows a user to provide input on the client machine that is then transmitted to the server. The input is executed on the user application running on the server. At step 2G the output of the user application (running on the server) is sent to the client machine, basically, as a picture of the server’s executable user application output. (See Paragraph 0098 of the Rice publication). That is, “the thin-client displays the file [output] . . . within a thin client window which contains the application’s graphical user interface.” (Paragraph 0108 of the Rice publication). This step is explained in more detail in the following paragraph of Rice, which states that the “application’s output [application running on the server] may then be sent as a “picture” by the server middleware to a web-enabling client . . .” (Paragraph 0109 of the Rice publication). Therefore, the Rice publication makes clear that during thin-client mode the user application runs on the server. The output of the user application is sent to the client device running the thin-client application (i.e., emulation program) that allows a user to see the output of the user application. This thin-client mode of operation however, requires the client device to remain connected to the server during use of the application.

**B. Fat/thick-client Mode Taught by the Rice Publication**

The Rice publication also teaches a mode of operation known as “fat” or “thick” client mode. The fat/thick-client mode of operation employed by the Rice publication, and relied upon by the Examiner is described in paragraph 0008 and paragraphs 0107-0109 of the Rice publication. The Applicant has provided the following flow chart to aid in the description of the operation of the system as described by the Rice publication.



**FIGURE 3 – Fat/Thick-Mode of Operation Taught by the Rice Publication**

In the second client/server relationship, the client device is described as a “fat” or “thick” client. A fat/thick client, in contrast with a thin client, is a device that actually downloads executable code for a particular application or program from the server, allowing the client device to run the application independent of the connection with the server. The first two steps of the fat/thick client mode are the same as the first two steps of the thin client mode discussed with respect to Figure 2 above. The Rice publication teaches operating in thin-client mode (step 3C) while executable code for the application being run on the server in thin-client mode is downloaded to the client device (step 3D). (See paragraph 0100). Following the download of the executable code for the user application, the client device can switch to fat/thick-client mode in which the client/device can execute the application independent of the connection to the server (step 3E).

IV. Summary of the Present Invention as Compared to the Rice Publication

The following table provides a summary of the operation of the present invention as compared with the two modes of operation taught by the Rice publication. The column labeled "Server Operations" describes the responsibilities of the server during the client/server transaction. The column labeled "Client/Server Communication" describes the communication between the server and the client, with the arrows indicating the direction the communication is traveling, during a client/server transaction. The column labeled "Client Operations" describes the responsibilities of the client during the client/server transaction.

	<u>Server Operations</u>	<u>Client/Server Communication</u>	<u>Client Operations</u>
<u>Present Invention</u>	<ul style="list-style-type: none"> <li>- Stores text files</li> </ul>	<ul style="list-style-type: none"> <li>- Text files downloaded from the server to the client→</li> </ul>	<ul style="list-style-type: none"> <li>- Stores application assembler that:               <ul style="list-style-type: none"> <li>- Retrieves program logic from the downloaded text files</li> <li>- Assembles a functioning application from the retrieved program logic</li> </ul> </li> <li>- Run the functioning application on the client</li> </ul>
<u>Rice - Thin-client Mode</u>	<ul style="list-style-type: none"> <li>- Stores thin-client application</li> <li>- Executes user application based on received instructions</li> <li>- Generates output of the user application as a "picture" file</li> </ul>	<ul style="list-style-type: none"> <li>- Download executable code for thin-client application to the client→</li> <li>← Instructions from user transmitted to the user application running on the server</li> <li>- Output of the user application as a picture file transmitted to the client →</li> </ul>	<ul style="list-style-type: none"> <li>- Execute the thin-client application on the client using downloaded executable code</li> <li>- User provides instructions for the user application running on the server using the thin-client application</li> <li>- Thin-client application displays output of the user application provided as a picture by the server</li> </ul>
<u>Rice - Fat/thick-client Mode</u>	<ul style="list-style-type: none"> <li>- Stores executable code for the user application run by the server during thin-client mode</li> </ul>	<ul style="list-style-type: none"> <li>- Download the executable code for the user application to the client→</li> </ul>	<ul style="list-style-type: none"> <li>- Run the user application on the client by executing the downloaded executable code</li> </ul>

**FIGURE 4 – Comparison of the Present Invention with the Rice publication**

V. Rice Does Not Anticipate Independent Claim 1 of the Present InventionA. Independent Claim 1

Independent claim 1 is reproduced below. Text appearing in bold highlights claim limitations not taught by the Rice publication.

1. A system for deploying applications over a distributed network to an Internet-enabled device for interacting with a server, the server being in communication with the distributed network and having text files containing application logic, the system comprising:

an **application assembler for storing on and running on the Internet-enabled device**, the application assembler for **downloading one or more text files from the server**, **retrieving program logic from each of the downloaded text files**, and **assembling the retrieved program logic into a functioning application and running the functioning application on the Internet-enabled device** regardless of whether the Internet-enabled device remains connected to the server.

B. Rice's Thin-Client Mode Does Not Anticipate Independent Claim 1

Nowhere in the above description does the Rice publication teach an **"application assembler for storing and running on the Internet-enabled device"** as required by independent claim 1. In fact, the Rice publication does not teach the use of an application assembler at all. The only application downloaded by the client device (operating in thin client mode) is the thin client application (shown by step 2C of FIG. 2). Preferably, the thin client application is an emulation program that displays the output of the user application, which runs on the server, to be displayed on the client device. The thin client application is NOT an application assembler because it does NOT assemble an application (i.e., it does not generate a functioning application, but rather acts to display output generated by an application running on the server).

In addition, nowhere in the above description does the Rice publication teach **"downloading one or more text files from the server"** as required by independent claim 1. The Examiner contends in the Advisory Action that "Rice teaches that an e-mail or a web page can have a hyperlink to a certain application file or document file, containing text, that is remotely stored on a server." (Advisory Action, third paragraph). The Advisory Action further states:



[W]hen the user or recipient clicks on the hyperlink, embedded information directs the client computer to a server computer that contains the text and user application files. Rice teaches that a thin-client is automatically downloaded to the user, and the application or program associated with the file is started on the remote application server. (Advisory Action, third paragraph).

Therefore, the Examiner maintains that the Rice publication teaches that a hyperlink that points to application files or documents files that may contain text. However, the application file or document files are NOT downloaded to the client device (i.e., Internet-enabled device). As Rice makes clear, only a thin-client is downloaded to the user, the thin-client representing an executable application that is run on the client device. The application or program associated with the application file is NOT downloaded to the client device, but is instead run on the remote application server. Therefore, the Rice publication does not teach **“downloading one or more text files from the server”** as required by independent claim 1.

In addition, the Rice publication does not teach **“retrieving program logic from each of the downloaded text files”** as required by independent claim 1. The Examiner contends in the Advisory Action that Rice teaches “when the user or recipient clicks on the hyperlink, embedded information directs the client computer to a server computer that contains the text and user application files.” The Examiner seems to indicate that the information embedded within the hyperlink is akin to retrieving program logic from each of the text files, wherein the hyperlink represents the text file. However, the hyperlink file in Rice acts as an address (e.g., an URL address) that directs the client computer to the server where the application or document (containing text) resides (shown in step 2B of FIG. 2). Therefore, the hyperlink does not include embedded program logic that can be extracted and assembled into a functioning application as required by independent claim 1.

In the alternative, the Examiner also seems to indicate that the application files and document files stored on the server are the text files that are downloaded to the client device. However, the Rice publication explicitly states that in thin-client mode “the application or program associated with the file is started on the remote application server computer 142 . . . [and] the specific file designated by the link is then opened by the user

application [running on the server].” Therefore, any application or documents to be run in thin-client mode are run on the server, NOT the client device. Therefore, the thin-client mode of the Rice publication does not teach or describe **“retrieving program logic from each of the downloaded text documents”** as required by independent claim 1.

Furthermore, the Rice publication does not teach **“assembling the retrieved program logic into a functioning application”** as required by independent claim 1. The fact that the Rice publication does not teach an application assembler indicates that the Rice publication does not teach assembling retrieved program logic into a functioning application. Assembling the retrieved program logic into a functioning application requires the application assembler to parse and organize the retrieved program logic, such as specifications of objects and scripts defining action to be taken in response to various user actions, such that a functioning or “virtual” application is provided to the user of the client device. Nowhere in the description of the Rice publication is there any teaching regarding assembling an application in this manner. In the Advisory Action, the Examiner relies on the following to teach this limitation:

Rice mentions that a hyperlink may contain an embedded information or URL of a protocol file which may execute a series of instructions to download middleware (assembler) to the client machine from which the hyperlink was accessed. Rice also teaches that the application’s output can be assembled into a “picture” by the middleware and sent to the web-enabling client located on the local machine. (Advisory Action, paragraph 3).

The Advisory Action incorrectly characterizes “middleware” as an assembler. Middleware is commonly used to describe an application that connects one software system to another software system. In the Rice publication, the middleware includes components installed on the server and client device that allow the output of the “application being run on the server . . . to be sent as a “picture” by the server middleware to a web-enabling client resident on the client local machine.” As described in step 2C of the flowchart describing thin-client operation of Rice, the middleware downloaded to the client device in Rice consists of executable code, preferably an emulation program, which runs on the client device. The emulation program allows the output of the application running on the

server to literally be sent as a picture that is displayed by the emulation program. Communicating the output of the application program as a picture that is “painted” or displayed by the client device is NOT the same as **“assembling the retrieved program logic into a functioning application”** as required by independent claim 1.

Furthermore, the thin-client mode of the Rice publication does not teach **“running the functioning application on the Internet-enabled device.”** The Rice publication makes clear that during thin-client mode, the application (described as the user application) is executed on the server, NOT on the Internet-enabled device or client.

Therefore, the description provided by the Rice publication with respect to operation in thin-client mode does not teach each and every limitation of independent claim 1.

C. Rice’s Fat/Thick-Client Mode Does Not Anticipate Independent Claim 1

The fat/thick-client mode taught by the Rice publication also fails to teach the limitations of independent claim 1. The Rice publication’s description of fat/thick mode of operation does not teach an **application assembler**. The Rice publication teaches downloading executable code of the application that runs on the server during the thin-client mode to the client device. (See step 3D of FIG. 3). That is, the same application that is run on the server during thin-client mode is downloaded onto the client during fat/thick-client mode. However, the executable code downloaded onto the client is not an application assembler, but is the application itself. Therefore, the Rice publication does not teach an application assembler.

Furthermore, the Rice publication does not teach **downloading one or more text files from the server** as required by independent claim 1. As described above, executable code for the user application downloaded to the client in fat/thick-client mode is not a text file (i.e., in human readable form). Therefore, the Rice publication does not teach the download of one or more text files from the server.

In addition, the Rice publication does not teach **retrieving program logic from each of the downloaded text files and assembling the retrieved program logic into a functioning application** as required by independent claim 1. As described above, the

executable code downloaded by the client device in fat/thick-client mode is in executable format. That is, it is capable of being run as an independent process on the client device. In contrast, the present invention requires assembling the retrieved program logic into a functioning or "virtual" application, which does not require the generation of executable code. The benefit of the present invention is in the size of the files that must be downloaded from the server in order to assemble a functioning application on the client device. The size of an executable file, depending on the application, may be quite large and therefore may require a significant amount of time to download and a significant amount of space to store. In contrast, the present invention does not require executable code to be downloaded, but only text files containing program logic that can be used by the application assembler to assemble a functioning application. The downloaded text files may be significantly smaller than the executable code that would otherwise be downloaded and stored to the client device.

Therefore, the description provided by the Rice publication with respect to operation in fat/thick-client mode does not teach each and every limitation of independent claim 1, and the Applicant respectfully submits that independent claim 1 is in condition for allowance.

VI. Rice Does Not Anticipate Independent Claim 8 of the Present Invention

A. Independent Claim 8

Independent claim 8 is reproduced below. Text appearing in bold highlights claim limitations not taught by the Rice publication.

8. A system for deploying an application over a network to an Internet-enabled device, the network having a server containing one or more application logic files, the application logic files containing embedded application logic relating to a computer program, the system comprising:

**a program assembler for storing on and running on the Internet-enabled device, the program assembler for downloading application logic files, retrieving embedded application logic from the application logic files, and building the computer program from the retrieved embedded application logic, and running the computer program on the Internet-enabled device.**

B. Rice's Thin-Client Mode Does Not Anticipate Independent Claim 8

As discussed with respect to independent claim 1, the thin-client mode taught by the Rice publication does not teach a **program assembler for storing on and running on the Internet-enabled device**. The program assembler is an application that runs on the Internet-enabled device that is capable of building a functioning application based on application logic files downloaded from a server. That is, applications are distributed in the present invention by allowing an Internet-enabled device to download application logic files. Embedded application logic is retrieved from the application logic files by a program assembler that uses the embedded application logic to build a functioning application. A number of functioning applications may be assembled and run on the Internet-enabled device by downloading the desired application logic files.

In contrast, the Rice publication teaches downloading a thin-client application (i.e., emulation program) that runs on the client device. As discussed with respect to independent claim 1, the thin-client application is NOT a program assembler because it does NOT assemble an application (i.e., it does not generate a functioning application, but rather acts to display output generated by an application running on the server).

Furthermore, the thin-client mode described by the Rice publication does not teach **“downloading application logic files, retrieving embedded application logic from the application logic files”** as required by independent claim 8. In contrast, the thin-client mode taught by the Rice publication teaches downloading the executable code for a thin-client application (that is, executable code for the emulation program) such that output of the application running on the server can be displayed on the client device. The download of executable code is NOT the same as the download of application logic files, because the application logic files CANNOT simply be executed on the client device. Rather, the application logic files are used by the program assembler to generate or build a functioning application that can be run on the Internet-enabled device.

Finally, the Rice publication does not teach **“building the computer program from the retrieved embedded application logic, and running the computer program on the Internet-enabled device”** as required by independent claim 1. Because the Rice publication does not teach a program assembler, the Rice publication does not teach building

a computer program from embedded application logic. As discussed with respect to independent claim 1, nowhere in the Rice publication is there any teaching regarding building an application. In thin-client mode, the Rice publication teaches downloading and running a thin-client application (for example, an emulation program) on the client device. The output of the application running on the server is displayed on the client device by the emulation program. However, the emulation program does not assemble a computer program. Therefore, the Rice publication does not teach “**building the computer program from the retrieved embedded application logic**” (downloaded from the server) on the client device or Internet-enabled device.

C. Rice’s Fat/Thick-Client Mode Does Not Anticipate Independent Claim 8.

The fat/thick-client mode taught by the Rice publication also fails to teach the limitations of independent claim 8. The fat/thick-client mode described in the Rice publication does NOT teach a **program assembler**. As discussed with respect to independent claim 1, nowhere in the Rice publication is a program assembler described. Furthermore, the fat/thick-client mode described in the Rice publication teaches downloading executable code of the application that runs on the server during the thin-client mode. (See step 3D of FIG. 3). This executable code downloaded onto the client in the fat/thick-client mode is not a program assembler, but is the executable code for the program itself. Therefore, the fat/thick client mode described by the Rice publication does not teach an **application assembler** as required by independent claim 8.

Furthermore, the fat/thick-client mode described in the Rice publication does not teach **downloading application logic files, retrieving embedded application logic from the application logic files** as required by independent claim 8. As described above, the fat/thick client mode taught by the Rice publication requires executable code to be downloaded to the client. In contrast, the application logic files downloaded by the Internet-enabled device (i.e., client device) are not executable code that can be run on the client device. Rather, the application logic files contain embedded application logic that is retrieved by the program assembler and used to build a computer program. Therefore, the fat/thick client mode described by the Rice publication does not teach downloading

application logic files, retrieving embedded application logic from the application logic files as required by independent claim 8.

In addition, the Rice publication does not teach **“building the computer program from the retrieved embedded application logic”** as required by independent claim 8. The Rice publication teaches downloading code in executable format from a server. That is, it does not need to be assembled before being run on the client device. In contrast, the present invention requires **building the computer program**. As discussed with respect to independent claim 1, the benefit of storing a program assembler on the client device is the size of the files (application logic files) that must be downloaded in order to build a computer program is much smaller than the executable code that must otherwise be downloaded in order to run the application. Therefore, the fat/thick-client mode taught by the Rice publication does not teach **“building the computer program from the retrieved embedded application logic”** as required by independent claim 8.

For the reasons outlined above, the Applicant respectfully requests that the rejection to this claim be withdrawn.

VII. Rice Does Not Anticipate Independent Claim 16 of the Present Invention

A. Independent Claim 16

Independent claim 16 is reproduced below. Text appearing in bold highlights claim limitations not taught by the Rice publication.

16. A method for deploying a computer program over a network, the method comprising:

storing and running a software module on a client device of a user;  
providing to the client device text files containing embedded application program logic for the software module, the text files containing embedded program logic for the computer program to the [installed software module upon request;  
running the computer program assembled from the embedded program logic on the client device; and  
enabling user interaction with the computer program running on the client device.

B. Rice's Thin-Client Mode Does not Anticipate Independent Claim 16

As discussed with respect to independent claim 1, the thin-client mode taught

by the Rice publication does not teach **“storing and running a software module on a client device”** wherein the software module installs a computer program on the client device based on **“text files containing embedded program logic.”** Therefore, the software module described in independent claim 16 is synonymous with the application assembler described with respect to independent claim 1. As discussed with respect to independent claim 1, the thin-client mode taught by the Rice publication does not teach an application, program or software module capable of generating or assembling a computer program. Rather, the Rice publication teaches downloading a thin-client application (i.e., emulation program) that allows the client device to display the result of an application running on a server. The thin-client application does NOT install a computer program on the client device.

In addition, the thin-client mode taught by the Rice publication does not teach **providing to the client device text files containing embedded application program logic** for the software module,” as required by independent claim 16. As discussed with respect to independent claim 1, the Rice publication does not teach providing text files to the client device that contain embedded application program logic. Rather, the thin-client mode taught by the Rice publication teaches downloading executable code (not a text file) for a thin-client application and receiving a “picture” of the output generated by the application running on the server that is then displayed by the thin-client application. (See Step 2C of FIG. 2). Neither the thin-client application or the picture file received from the server is a text file, nor does either contain embedded application program logic that is used by the software module to install a computer program on the client device.

Finally, the Rice publication does not teach **“running the computer program assembled from the embedded program logic on the client device”** as required by independent claim 16. The Rice publication teaches downloading the executable code for the thin-client application to the client device. That is, the thin-client application is downloaded in a form that allows it to be run immediately. In addition, the Rice publication makes clear that during thin-client mode, the user application program is run *on the server, not on the client device* (shown in Step 2D above, and paragraph 0107 of the Rice publication). Therefore, the Rice publication does not teach **“running the computer program assembled from the embedded program logic on the client device”** as required by independent cl. 16.



C. Rice's Fat/Thick-Client Mode Does Not Anticipate Independent Claim 16.

The fat/thick-client mode taught by the Rice publication also fails to teach each and every limitation of independent claim 16. The Rice publication's description of fat/thick mode of operation does not teach the step of **"providing to the client device text files containing embedded application program logic** for the software module" as required by independent claim 16. As discussed with respect to independent claim 1, the fat/thick client mode described by the Rice publication teaches downloading executable code of the application that runs on the server during the thin-client mode. That is, the same application that is run on the server during thin-client mode is downloaded onto the client during fat/thick-client mode. However, the executable code downloaded onto the client is not a text file and does not contain embedded application program logic. Therefore, the Rice publication fails to teach the step of **"providing to the client device text files containing embedded application program logic."**

Finally, the Rice publication does not teach **"running the computer program assembled from the embedded program logic on the client device"** as required by independent claim 16. The fat/thick-client mode described by the Rice publication teaches downloading the executable code for the application to be run on the client device. That is, the executable code for the application that runs on the server during thin-client mode is downloaded to the client for fat/thick-client mode. Because the application downloaded from the server is executable code, it does not need to be assembled at the client device. This is in contrast with independent claim 16, which requires **"running the computer program assembled from the embedded program logic on the client device."**

For the reasons outlined above, the Rice publication does not teach each and every element of independent claim 16. The Applicant respectfully requests that the rejection to this claim be withdrawn.

VIII. Dependent claims 2-5, 9-15, 17, and 19-22

Claims 2-5, 8-15, 17, and 19-22 depend respectively from independent claims 1, 8, and 16, and are allowable therewith. In addition, the combinations of features recited in claims 2-5, 8-15, 17, and 19-22 are independently patentable, although this does not need to

be specifically addressed herein since any claim depending from a patentable independent claim is also patentable. See M.P.E.P. 2143.03, citing *In re Fine*, 5 U.S.P.Q.2d (BNA) 1596 (Fed. Cir. 1988).

In particular, the Examiner states that the limitations of dependent claims 3 and 12, which require as a limitation that the **“program logic is operating system independent,”** are taught by the Rice publication. The Examiner relies on this paragraph from the Rice publication to teach that the embedded application logic is operating system independent:

Furthermore, the platform, operating system, or resident applications of the recipient machine are immaterial. The recipient 620 of FIG. 6 could be running on a mainframe, a Unix server, a Microsoft Windows server, or other machine with a compatible thin client. Because all that is required on the recipient's machine is the relatively limited processing power required to run the thin client [application], the recipient may have virtual access to a relatively vast amount of processing capability – the application [i.e., server side] could be running on various configurations, for example, a mainframe, parallel multiprocessor machine, or Unix cluster supercomputer.” (Paragraph 0145 of the Rice publication)

Nothing in this paragraph indicates that Rice teaches embedded application logic that is platform independent. In fact, the statement that the recipient (i.e., client) could be running on a mainframe, a Unix server, a Microsoft Windows server, or other machine with a *compatible* thin client” indicates that the thin-client software (preferable emulation software) is NOT platform independent, but must be *compatible* with the type of platform on which it is running. In addition, the point being made by this paragraph is that any type of device may serve as the recipient or client device, regardless of computing power. Because processing of an application is done on the server, which may contain vast computing power, the client device may have “meager computing power.” Therefore, the rejection of dependent claims 3 and 12 should also be withdrawn.

IX. The Combination of the Rice Publication and the Lloyd Patent Do NOT  
Render Claims 6 and 7 Obvious

Claims 6 and 7 depend from independent claim 1, and are allowable therewith. In addition, it is respectfully submitted that the combinations of features recited in claims 6 and 7 are independently patentable, although this does not need to be specifically addressed herein since any claim depending from a patentable independent claim is also patentable. *See* M.P.E.P. 2143.03, citing *In re Fine*, 5 U.S.P.Q.2d (BNA) 1596 (Fed. Cir. 1988).

Specifically, claim 6 recites **“a script engine for interpreting scripts contained in the extracted program logic, and for providing methods to invoke script functions.”** With respect to claim 6, the Examiner states that the Lloyd patent teaches a script engine for interpreting scripts contained in the extracted program logic. The Lloyd patent recites that a “web software 404 also includes Common Gateway Interface (CGI) scripts 405 to create dynamic HTML files. *A CGI script is an external application executed by a web server. A CGI script is invoked when a user submits a request for dynamic information.*” (Lloyd, Col. 9, ll. 27-40).

As described by the above paragraph, the Lloyd reference describes a script engine that is *executed by a web server*. The script engine is only invoked when a user submits a request to the web server for dynamic information. This is in contrast with the present invention, in which the script engine is located locally on the Internet-enabled device (i.e., client). Because the Lloyd reference does not describe a script engine that operates on an Internet-enabled device (i.e., client), the combination of the Lloyd patent and the Rice publication does not teach each and every limitation of dependent claim 6.


X. Conclusion

In view of the foregoing, it is respectfully requested that the appeal of claims 1-17 and 19-22 be granted, such that pending claims 1-17 and 19-22 of this application are allowed.

Respectfully submitted,

KINNEY & LANGE, P.A.

Date: 10/2/06

By:   
Michael A. Collins, Reg. No. 59,135  
THE KINNEY & LANGE BUILDING  
312 South 3rd Street  
Minneapolis, MN 55402-1002  
Telephone: (612) 339-1863  
Fax: (612) 339-6580

### **Claims Appendix**

1.(Previously Presented) A system for deploying applications over a distributed network to an Internet-enabled device for interacting with a server, the server being in communication with the distributed network and having text files containing application logic, the system comprising:

an application assembler for storing on and running on the Internet-enabled device, the application assembler for downloading one or more text files from the server, retrieving program logic from each of the downloaded text files, and assembling the retrieved program logic into a functioning application and running the functioning application on the Internet-enabled device regardless of whether the Internet-enabled device remains connected to the server.

2. (Original) The system of claim 1, wherein the application assembler is operating system dependent.

3. (Previously Presented) The system of claim 1, wherein the program logic is operating system independent.

4. (Previously Presented) The system of claim 1, wherein the functioning application provides a graphical user interface for receiving and interpreting user inputs to the Internet-enabled device.

5. (Previously Presented) The system of claim 4, wherein the functioning application processes the user inputs and interacts with a local or remote database, or both, for performing user instructions.

6. (Original) The system of claim 1, the application assembler comprising:  
a parser for extracting program logic from text files stored on the server;  
a script engine for interpreting scripts contained in the extracted program

logic, and for providing methods to invoke script functions; and component handlers for rendering visual components and for processing operations specific to the visual components.

7. (Original) The system of claim 6, wherein the application assembler further comprises:  
a layout handler for analyzing positioning properties of a group of components and translating them into component dimensions and coordinates for display on each web enabled device.

8. (Previously Presented) A system for deploying an application over a network to an Internet-enabled device, the network having a server containing one or more application logic files, the application logic files containing embedded application logic relating to a computer program, the system comprising:

a program assembler for storing on and running on the Internet-enabled device, the program assembler for downloading application logic files, retrieving embedded application logic from the application logic files, and building the computer program from the retrieved embedded application logic, and running the computer program on the Internet-enabled device.

9. (Original) The system of claim 8, further comprising:

a plugin for installation in a web-browser for running the program assembler according to instructions embedded in an Internet web page.

10. (Original) The system of claim 8, wherein the program assembler is operating system dependent, the program assembler for assembling multiple computer programs based on the embedded application logic.

11. (Previously Presented) The system of claim 8, wherein the program assembler is

operating system dependent, and wherein at least two different program assemblers for at least two different operating systems on two different Internet-enabled devices use the embedded application logic from the text files for building a computer program having the same functionality on both Internet-enabled devices.

12. (Original) The system of claim 8, wherein the embedded application logic is operating system independent.

13. (Previously Presented) The system of claim 8, wherein the computer program provides a graphical user interface for receiving and interpreting user inputs to the Internet-enabled device.

14. (Previously Presented) The system of claim 8, wherein the Internet-enabled device is selected from a group consisting of computers, workstations, personal digital assistants, wireless personal digital assistants, and Internet-enabled phones.

15. (Original) The system of claim 8, wherein the application logic files are compressed.

16. (Previously Presented) A method for deploying a computer program over a network, the method comprising:

storing and running a software module on a client device of a user;

providing to the client device text files containing embedded application program logic for the software module, the text files containing embedded program logic for the computer program to the installed software module upon request;

running the computer program assembled from the embedded program logic on the client device; and

enabling user interaction with the computer program running on the client device.

17. (Original) The method of claim 16, wherein the step of hosting comprises:  
storing a compressed file in a standard compression format on a server in  
communication with a network, the compressed file for automatic  
download and installation on the client device through a web browser.
18. (Canceled)
19. (Original) The method of claim 16, wherein the step of providing text files comprises:  
storing text files on a server in communication with a network, the text files  
containing embedded program logic.
20. (Original) The method of claim 19, wherein the text files are compressed.
21. (Previously Presented) The method of claim 16, and further comprising:  
hosting a web page containing a software module and a plugin on for  
installation on a client device of a user; and  
launching the installed software module using the installed plugin based on  
instructions embedded within the web page.
22. (New) The method of claim 21, wherein the step of launching the installed software  
module comprises:  
embedding a launch instruction in a starter web page on the network.



**Evidence Appendix**

1. Evidence entered by the Examiner and relied upon by the Appellant:  
None.
  
2. Evidence relied upon by the Examiner as to grounds of rejection to be reviewed on appeal:
  - a. Rice III, U.S. Pub. US2002/0174010 A1 (entered into the record by the Examiner in the August 15, 2005 Office Action).
  - b. Lloyd, U.S. Patent No. 6,779,178 ("the Lloyd patent") (entered into the record by the Examiner in the August 15, 2005 Office Action).

First Named Inventor: Timothy J. Block

Application No.: 10/081,921

-31-

**Related Proceedings Appendix**

None.

First Named Inventor: Timothy J. Block

Application No.: 10/081,921

-32-

**Table of Cases Appendix**

1. *In re Fine*, 5 U.S.P.Q.2d (BNA) 1596 (Fed. Cir. 1988).